Amazon SDE Internship — 1-Day OA Practice: 40 Problems + C++ Solutions

Instructions: Solve each problem, then compare with the provided concise C++ solution. Use these to simulate an OA: time yourself (2 coding questions at medium difficulty in 90–120 minutes, plus debugging & workstyle).

1. Two Sum (Array, Hashmap): Given array nums and target, return indices of two numbers that add up to target.

```
#include <bits/stdc++.h> using namespace std; int
main(){ios::sync_with_stdio(false);cin.tie(NULL);     int n; if(!(cin>>n)) return 0;
vector<int>a(n); for(int i=0;i<n;i++)cin>>a[i];     int target; cin>>target;
unordered_map<int,int> mp;     for(int i=0;i<n;i++){int need=target-a[i];
if(mp.count(need)){cout<<mp[need]<<" "<<i<<"\n"; return 0;} mp[a[i]] = i;}     cout<<"-1 -1\n";
}
```

2. Reverse String (Two pointers): Reverse a string in-place.

```
#include <bits/stdc++.h> using namespace std; int main(){string s; getline(cin,s); int
i=0,j=s.size()-1; while(i<j) swap(s[i++],s[j--]); cout<<s<<"\n"; }
```

3. Longest Substring Without Repeating Characters (Sliding window): Return length of longest substring without repeating chars.

```
#include <bits/stdc++.h> using namespace std; int main(){string s; getline(cin,s); vector<int>
last(256,-1); int start=0,ans=0; for(int i=0;i<s.size();++i){ if(last[s[i]]>=start)
start=last[s[i]]+1; ans=max(ans,i-start+1); last[s[i]]=i;} cout<<ans<<"\n"; }
```

4. Merge Intervals (Sorting): Given intervals, merge overlapping ones.

```
#include <bits/stdc++.h> using namespace std; int main(){int n; if(!(cin>>n)) return 0;
vector<pair<int,int>> a(n); for(int i=0;i<n;i++) cin>>a[i].first>>a[i].second; sort(a.begin(),
a.end()); vector<pair<int,int>> res; for(auto &iv: a){
if(res.empty()||res.back().second<iv.first) res.push_back(iv); else res.back().second =
max(res.back().second, iv.second); } for(auto &r:res) cout<<r.first<<" "<<r.second<<"\n"; }
```

5. Valid Parentheses (Stack): Check if parentheses string is valid.

```
#include <bits/stdc++.h> using namespace std; bool valid(string s){ stack<char> st; for(char
c:s){ if(c=='('||c=='['||c=='{') st.push(c); else{ if(st.empty()) return false; char
t=st.top(); st.pop();  if((c==')'&&t!='(')||(c==']'&&t!='[')||(c=='}'&&t!='{')) return false;
}} return st.empty(); } int main(){string s; getline(cin,s); cout<<(valid(s)?
"YES":"NO")<<"\n"; }
```

6. Binary Search (Classic): Search target in sorted array; return index or -1.

```
#include <bits/stdc++.h> using namespace std; int bs(vector<int>&a,int x){ int
l=0,r=a.size()-1; while(l<=r){int m=l+(r-l)/2; if(a[m]==x) return m; if(a[m]<x) l=m+1; else
r=m-1;} return -1;} int main(){int n,x; if(!(cin>>n)) return 0; vector<int>a(n); for(int
i=0;i<n;i++)cin>>a[i]; cin>>x; cout<<bs(a,x)<<"\n"; }
```

7. Search in Rotated Sorted Array: Find target in rotated sorted array (no duplicates).

```
#include <bits/stdc++.h> using namespace std; int searchRot(vector<int>&a,int t){ int
l=0,r=a.size()-1; while(l<=r){int m=(l+r)/2; if(a[m]==t) return m;  if(a[l]<=a[m]){ if(a[l]<=t
&& t<a[m]) r=m-1; else l=m+1; } else{ if(a[m]<t && t<=a[r]) l=m+1; else r=m-1; } } return -1;}
int main(){int n; if(!(cin>>n)) return 0; vector<int>a(n); for(int i=0;i<n;i++)cin>>a[i]; int
t;cin>>t; cout<<searchRot(a,t)<<"\n"; }
```

8. Two Pointers - 3Sum (Simplified): Return unique triplets summing to zero.

```
#include <bits/stdc++.h> using namespace std; int main(){int n; if(!(cin>>n)) return 0;
vector<int>a(n); for(int i=0;i<n;i++)cin>>a[i];  sort(a.begin(),a.end()); set<vector<int>> ans;
for(int i=0;i<n;i++){ int l=i+1,r=n-1; while(l<r){int s=a[i]+a[l]+a[r];
if(s==0){ans.insert({a[i],a[l],a[r]}); l++; r--; } else if(s<0) l++; else r--; }} for(auto
v:ans){ for(int x:v) cout<<x<<" "; cout<<"\n"; } }
```

9. Maximum Subarray (Kadane): Return max subarray sum.

```
#include <bits/stdc++.h> using namespace std; int main(){int n; if(!(cin>>n)) return 0; long
long ans=LLONG_MIN, cur=0; for(int i=0;i<n;i++){ long long x; cin>>x; cur=max(x,cur+x);
ans=max(ans,cur);} cout<<ans<<"\n"; }
```

10. Container With Most Water (Two pointers): Max area between lines.

```
#include <bits/stdc++.h> using namespace std; int main(){int n; if(!(cin>>n)) return 0;
vector<int>h(n); for(int i=0;i<n;i++)cin>>h[i];  int l=0,r=n-1; long long best=0; while(l<r){
best=max(best, (long long)(r-l)*min(h[l],h[r])); if(h[l]<h[r]) l++; else r--; }
cout<<best<<"\n"; }
```

11. Merge Two Sorted Lists (Linked List): Merge two sorted linked lists into one list (print merged values).

```
#include <bits/stdc++.h> using namespace std; int main(){int n,m; if(!(cin>>n>>m)) return 0;
vector<int>a(n),b(m); for(int i=0;i<n;i++)cin>>a[i]; for(int j=0;j<m;j++)cin>>b[j];
vector<int>res; int i=0,j=0; while(i<n && j<m){ if(a[i]<b[j]) res.push_back(a[i++]); else
res.push_back(b[j++]); } while(i<n) res.push_back(a[i++]); while(j<m) res.push_back(b[j++]);
for(int x:res) cout<<x<<" "; cout<<"\n"; }
```

12. Linked List Cycle (Floyd): Given array where value is next index or -1, detect cycle existence.

```
#include <bits/stdc++.h> using namespace std; int main(){int n; if(!(cin>>n)) return 0;
vector<int>a(n); for(int i=0;i<n;i++)cin>>a[i];  int slow=0, fast=0; auto nxt=[&](int i){
if(i==-1) return -1; return a[i]; };  while(true){ slow = nxt(slow); if(fast==-1) break; fast =
nxt(fast); if(fast==-1) break; fast = nxt(fast); if(slow==-1 || fast==-1) break;
if(slow==fast){ cout<<\"Cycle\\n\"; return 0;} } cout<<\"No Cycle\\n\"; }
```

13. Binary Tree Inorder Traversal (DFS): Given n and list of nodes with left/right indices, print inorder.

```
#include <bits/stdc++.h> using namespace std; void inorder(int u, vector<int>&val,
vector<int>&l, vector<int>&r){  if(u==-1) return; inorder(l[u],val,l,r); cout<<val[u]<<" ";
inorder(r[u],val,l,r); } int main(){int n; if(!(cin>>n)) return 0; vector<int>val(n),L(n),R(n);
for(int i=0;i<n;i++)cin>>val[i]>>L[i]>>R[i]; inorder(0,val,L,R); cout<<"\n"; }
```

14. BFS Shortest Path (Unweighted): Shortest path length in unweighted graph from src to dest.

```
#include <bits/stdc++.h> using namespace std; int main(){int n,m; if(!(cin>>n>>m)) return 0;
vector<vector<int>> g(n);  for(int i=0;i<m;i++){int u,v;cin>>u>>v; g[u].push_back(v);
g[v].push_back(u);}  int s,t; cin>>s>>t; vector<int>d(n,-1); queue<int>q; d[s]=0; q.push(s);
while(!q.empty()){int u=q.front();q.pop(); for(int v:g[u]) if(d[v]==-1){ d[v]=d[u]+1;
q.push(v); }}  cout<<d[t]<<"\n"; }
```

15. Lowest Common Ancestor (Binary Lifting idea simplified): Find LCA by parent pointers using visited set (works if parent pointers exist).

```
#include <bits/stdc++.h> using namespace std; int main(){int n; if(!(cin>>n)) return 0;
vector<int>par(n); for(int i=0;i<n;i++)cin>>par[i];  int a,b; cin>>a>>b; unordered_set<int>s;
while(a!=-1){ s.insert(a); a=par[a]; } while(b!=-1){ if(s.count(b)){ cout<<b<<"\n"; return 0;}
b=par[b]; } cout<<"-1\n"; }
```

16. Permutations (Backtracking): Generate all permutations of n numbers.

```
#include <bits/stdc++.h> using namespace std; void back(vector<int>&a, vector<int>&cur,
vector<bool>&vis){  if(cur.size()==a.size()){ for(int x:cur) cout<<x<<" "; cout<<"\n"; return;
}  for(int i=0;i<a.size();++i) if(!vis[i]){ vis[i]=true; cur.push_back(a[i]); back(a,cur,vis);
cur.pop_back(); vis[i]=false; } } int main(){int n; if(!(cin>>n)) return 0; vector<int>a(n);
for(int i=0;i<n;i++)cin>>a[i]; vector<bool>vis(n,false); vector<int>cur; back(a,cur,vis); }
```

17. Subsets (Bitmask): Print all subsets of an array.

```
#include <bits/stdc++.h> using namespace std; int main(){int n; if(!(cin>>n)) return 0;
vector<int>a(n); for(int i=0;i<n;i++)cin>>a[i];  for(int mask=0; mask<(1<<n); ++mask){ for(int
i=0;i<n;++i) if(mask&(1<<i)) cout<<a[i]<<" "; cout<<"\n"; } }
```

18. Coin Change (DP - ways): Count ways to make amount with coins.

```
#include <bits/stdc++.h> using namespace std; int main(){int m,n; if(!(cin>>m>>n)) return 0;
vector<int> coins(m); for(int i=0;i<m;i++)cin>>coins[i];  vector<long long> dp(n+1,0); dp[0]=1;
for(int c:coins) for(int x=c;x<=n;++x) dp[x]+=dp[x-c]; cout<<dp[n]<<"\n"; }
```

19. Longest Increasing Subsequence (n log n): Length of LIS in array.

```
#include <bits/stdc++.h> using namespace std; int main(){int n; if(!(cin>>n)) return 0;
vector<int>a(n); for(int i=0;i<n;i++)cin>>a[i];  vector<int>d; for(int x:a){ auto
it=lower_bound(d.begin(),d.end(),x); if(it==d.end()) d.push_back(x); else *it=x; }
cout<<d.size()<<"\n"; }
```

20. Top K Frequent Elements (Heap/Hash): Return k most frequent elements.

```
#include <bits/stdc++.h> using namespace std; int main(){int n,k; if(!(cin>>n)) return 0;
vector<int>a(n); for(int i=0;i<n;i++)cin>>a[i]; cin>>k;  unordered_map<int,int>cnt; for(int
x:a) cnt[x]++;  priority_queue<pair<int,int>>pq; for(auto &p:cnt) pq.push({p.second,p.first});
while(k-- && !pq.empty()){ cout<<pq.top().second<<" "; pq.pop(); } cout<<"\n"; }
```

21. Kth Smallest Element in Matrix (min-heap): Find kth smallest in sorted rows matrix (simple push all for brevity).

```
#include <bits/stdc++.h> using namespace std; int main(){int n,k; if(!(cin>>n>>k)) return 0;
vector<int> a; for(int i=0;i<n*n;i++){int x;cin>>x;a.push_back(x);} sort(a.begin(),a.end());
cout<<a[k-1]<<"\n"; }
```

22. Validate Binary Search Tree (Inorder): Check if tree array is BST via inorder previous check.

```
#include <bits/stdc++.h> using namespace std; bool ok=true; long long prevv=LLONG_MIN; void
inorder(int u, vector<int>&val, vector<int>&L, vector<int>&R){  if(u==-1) return;
inorder(L[u],val,L,R); if(val[u]<=prevv) ok=false; prevv=val[u]; inorder(R[u],val,L,R); } int
main(){int n; if(!(cin>>n)) return 0; vector<int>val(n),L(n),R(n); for(int
i=0;i<n;i++)cin>>val[i]>>L[i]>>R[i]; inorder(0,val,L,R); cout<<(ok?\"YES\":\"NO\")<<\"\\n\"; }
```

23. Permutation in String (Sliding window + freq): Check if s2 contains permutation of s1.

```
#include <bits/stdc++.h> using namespace std; int main(){string s1,s2; if(!(cin>>s1>>s2))
return 0; vector<int>f(26,0);  for(char c:s1) f[c-'a']++; int l=0,match=0;  for(int
r=0;r<s2.size();++r){ int idx=s2[r]-'a'; if(--f[idx]==0) match++; if(r-l+1> s1.size()){
if(++f[s2[l]-'a']==0) match--; l++; } if(match==26 || r-l+1==s1.size() &&
accumulate(f.begin(),f.end(),0)==0) { cout<<\"YES\\n\"; return 0;} } cout<<\"NO\\n\"; }
```

24. Group Anagrams (Hash map with sorted key): Group anagram strings.

```
#include <bits/stdc++.h> using namespace std; int main(){int n; if(!(cin>>n)) return 0;
vector<string>s(n); for(int i=0;i<n;i++)cin>>s[i];  unordered_map<string,vector<string>> mp;
for(auto &w:s){ string t=w; sort(t.begin(),t.end()); mp[t].push_back(w); }  for(auto &p:mp){
for(auto &x:p.second) cout<<x<<" "; cout<<"\n"; } }
```

25. Number of Islands (DFS): Count islands (grid of 0/1).

```
#include <bits/stdc++.h> using namespace std; void dfs(int i,int j,vector<string>&g){
if(i<0||j<0||i>=g.size()||j>=g[0].size()||g[i][j]=='0') return; g[i][j]='0'; dfs(i+1,j,g);
dfs(i-1,j,g); dfs(i,j+1,g); dfs(i,j-1,g); } int main(){int r,c; if(!(cin>>r>>c)) return 0;
vector<string>g(r); for(int i=0;i<r;i++)cin>>g[i];  int cnt=0; for(int i=0;i<r;i++) for(int
j=0;j<c;j++) if(g[i][j]=='1'){ cnt++; dfs(i,j,g); } cout<<cnt<<"\n"; }
```

26. Word Ladder (BFS - shortest transformation): Find number of steps from start to end transforming one letter at a time using word list.

```
#include <bits/stdc++.h> using namespace std; int main(){ string begin,end; int n;
if(!(cin>>begin>>end>>n)) return 0; unordered_set<string>dict; string w; for(int
i=0;i<n;i++){cin>>w; dict.insert(w);} queue<pair<string,int>>q; q.push({begin,1});
while(!q.empty()){ auto [s,d]=q.front(); q.pop(); if(s==end){ cout<<d<<"\n"; return 0;} for(int
i=0;i<s.size();++i){ string t=s; for(char c='a';c<='z';++c){ t[i]=c; if(dict.count(t)){
dict.erase(t); q.push({t,d+1}); } } } } cout<<\"0\\n\"; }
```

27. Serialize/Deserialize (Basic idea): Simple level-order serialization to vector and back (demonstrative).

```
#include <bits/stdc++.h> using namespace std; int main(){ // omitted detailed I/O for brevity -
concept shown  cout<<\"Serialize/deserialize concept: use BFS with null markers (implementation
depends on format).\\n\"; }
```

28. Product of Array Except Self (Prefix/Suffix): Return array where each position is product of all other elements (no division).

```
#include <bits/stdc++.h> using namespace std; int main(){int n; if(!(cin>>n)) return 0;
vector<long long>a(n); for(int i=0;i<n;i++)cin>>a[i];  vector<long
long>pref(n,1),suf(n,1),res(n);  for(int i=1;i<n;i++) pref[i]=pref[i-1]*a[i-1];  for(int
i=n-2;i>=0;i--) suf[i]=suf[i+1]*a[i+1];  for(int i=0;i<n;i++) res[i]=pref[i]*suf[i]; for(auto
x:res) cout<<x<<\" \"; cout<<\"\\n\"; }
```

29. Find Duplicate Number (Floyd cycle): Find duplicate in array of n+1 numbers in [1..n]

```
#include <bits/stdc++.h> using namespace std; int main(){int n; if(!(cin>>n)) return 0;
vector<int>a(n); for(int i=0;i<n;i++)cin>>a[i];  int slow=a[0], fast=a[0]; do{ slow=a[slow];
fast=a[a[fast]]; } while(slow!=fast); fast=a[0]; while(slow!=fast){ slow=a[slow]; fast=a[fast];
} cout<<slow<<\"\\n\"; }
```

30. Sliding Window Maximum (Deque): Return max in each sliding window of size k.

```
#include <bits/stdc++.h> using namespace std; int main(){int n,k; if(!(cin>>n>>k)) return 0;
vector<int>a(n); for(int i=0;i<n;i++)cin>>a[i];  deque<int>dq; for(int i=0;i<n;i++){
while(!dq.empty() && dq.front()<=i-k) dq.pop_front(); while(!dq.empty() && a[dq.back()]<=a[i])
dq.pop_back(); dq.push_back(i); if(i>=k-1) cout<<a[dq.front()]<<\" \"; } cout<<\"\\n\"; }
```

31. Minimum Window Substring (Sliding window): Find minimum window in s covering all chars of t.

```
#include <bits/stdc++.h> using namespace std; int main(){string s,t; if(!(cin>>s>>t)) return 0;
vector<int>need(128,0); for(char c:t) need[c]++;  int missing=t.size(),l=0,
start=0,len=INT_MAX; for(int r=0;r<s.size();++r){ if(--need[s[r]]>=0) missing--;
while(missing==0){ if(r-l+1<len){ start=l; len=r-l+1; } if(++need[s[l]]>0) missing++; l++; } }
if(len==INT_MAX) cout<<\"\"<<\"\\n\"; else cout<<s.substr(start,len)<<\"\\n\"; }
```

32. Decode Ways (DP): Count decodings of digit string.

```
#include <bits/stdc++.h> using namespace std; int main(){string s; if(!(cin>>s)) return 0; int
n=s.size(); vector<int>dp(n+1,0); dp[0]=1;  for(int i=1;i<=n;i++){ if(s[i-1]!='0')
dp[i]+=dp[i-1]; if(i>1){ int two = (s[i-2]-'0')*10 + (s[i-1]-'0'); if(two>=10 && two<=26)
dp[i]+=dp[i-2]; } } cout<<dp[n]<<"\n"; }
```

33. Partition Equal Subset Sum (DP subset sum): Can array be partitioned into two equal-sum subsets?

```
#include <bits/stdc++.h> using namespace std; int main(){int n; if(!(cin>>n)) return 0;
vector<int>a(n); int sum=0; for(int i=0;i<n;i++){cin>>a[i]; sum+=a[i];}  if(sum%2){
cout<<\"NO\\n\"; return 0;} int target=sum/2; vector<char>dp(target+1,0); dp[0]=1; for(int x:a)
for(int v=target;v>=x;--v) if(dp[v-x]) dp[v]=1; cout<<(dp[target]?\"YES\":\"NO\")<<\"\\n\"; }
```

34. Rotate Image (Matrix transpose+reverse): Rotate n x n matrix by 90 degrees clockwise in-place.

```
#include <bits/stdc++.h> using namespace std; int main(){int n; if(!(cin>>n)) return 0;
vector<vector<int>>a(n,vector<int>(n)); for(int i=0;i<n;i++)for(int j=0;j<n;j++)cin>>a[i][j];
for(int i=0;i<n;i++) for(int j=i+1;j<n;j++) swap(a[i][j],a[j][i]);  for(int i=0;i<n;i++)
reverse(a[i].begin(),a[i].end());  for(auto &row:a){ for(int x:row) cout<<x<<\" \";
cout<<\"\\n\"; } }
```

35. Sort Colors (Dutch National Flag): Sort array of 0s,1s,2s in one pass.

```
#include <bits/stdc++.h> using namespace std; int main(){int n; if(!(cin>>n)) return 0;
vector<int>a(n); for(int i=0;i<n;i++)cin>>a[i];  int low=0,mid=0,high=n-1; while(mid<=high){
if(a[mid]==0) swap(a[low++],a[mid++]); else if(a[mid]==1) mid++; else swap(a[mid],a[high--]); }
for(int x:a) cout<<x<<\" \"; cout<<\"\\n\"; }
```

36. Find Median from Data Stream (Heaps): Maintain running median with two heaps.

```
#include <bits/stdc++.h> using namespace std; int main(){int n; if(!(cin>>n)) return 0;
priority_queue<int> lo; priority_queue<int,vector<int>,greater<int>> hi; for(int
i=0;i<n;i++){int x;cin>>x; if(lo.empty()||x<=lo.top()) lo.push(x); else hi.push(x);
if(lo.size()>hi.size()+1){ hi.push(lo.top()); lo.pop(); } else if(hi.size()>lo.size()){
lo.push(hi.top()); hi.pop(); } double med = lo.size()==hi.size()? (lo.top()+hi.top())/2.0:
lo.top(); cout<<med<<\"\\n\"; } }
```

37. Word Search (Backtracking on grid): Check if word exists in board by adjacent letters (no reuse).

```
#include <bits/stdc++.h> using namespace std; int R,C; vector<string>b; string w; bool dfs(int
i,int j,int idx, vector<vector<char>>&vis){  if(idx==w.size()) return true;
if(i<0||j<0||i>=R||j>=C||vis[i][j]||b[i][j]!=w[idx]) return false; vis[i][j]=1;  bool ok =
dfs(i+1,j,idx+1,vis)||dfs(i-1,j,idx+1,vis)||dfs(i,j+1,idx+1,vis)||dfs(i,j-1,idx+1,vis);
vis[i][j]=0; return ok; } int main(){int r,c; if(!(cin>>r>>c)) return 0; R=r; C=c; b.resize(r);
for(int i=0;i<r;i++) cin>>b[i]; cin>>w; vector<vector<char>>vis(r, vector<char>(c,0));
cout<<(dfs(0,0,0,vis)?\"YES\":\"NO\")<<\"\\n\"; }
```

38. Maximum Product Subarray: Max product of any subarray (handles negatives).

```
#include <bits/stdc++.h> using namespace std; int main(){int n; if(!(cin>>n)) return 0;
vector<long long>a(n); for(int i=0;i<n;i++)cin>>a[i];  long long ans=LLONG_MIN, curmax=1,
curmin=1;  for(long long x:a){ if(x==0){ curmax=1; curmin=1; ans=max(ans,0LL); continue;} long
long tmp=curmax*x; curmax=max({x, tmp, curmin*x}); curmin=min({x, tmp, curmin*x});
ans=max(ans,curmax); } cout<<ans<<\"\\n\"; }
```

39. Find First Missing Positive (Index mapping): Find smallest missing positive integer.

```
#include <bits/stdc++.h> using namespace std; int main(){int n; if(!(cin>>n)) return 0;
vector<int>a(n); for(int i=0;i<n;i++)cin>>a[i];  for(int i=0;i<n;i++){ while(a[i]>0 && a[i]<=n
&& a[a[i]-1]!=a[i]) swap(a[i], a[a[i]-1]); }  for(int i=0;i<n;i++) if(a[i]!=i+1){
cout<<i+1<<\"\\n\"; return 0; } cout<<n+1<<\"\\n\"; }
```

40. LRU Cache (Hash+DLL simulation via list and map): Implement LRU cache with get/put (simple demo printing state).

```
#include <bits/stdc++.h> using namespace std; class LRU{  int cap; list<pair<int,int>> lst;
unordered_map<int, list<pair<int,int>>::iterator> mp; public:  LRU(int c):cap(c){}  int get(int
k){ if(!mp.count(k)) return -1; auto it=mp[k]; int v=it->second; lst.erase(it);
lst.push_front({k,v}); mp[k]=lst.begin(); return v; }  void put(int k,int v){ if(mp.count(k)){
lst.erase(mp[k]); mp.erase(k);} if(lst.size()==cap){ mp.erase(lst.back().first);
lst.pop_back(); } lst.push_front({k,v}); mp[k]=lst.begin(); } }; int main(){int n;
if(!(cin>>n)) return 0; int cap; cin>>cap; LRU l(cap); for(int i=0;i<n;i++){ string cmd;
cin>>cmd; if(cmd==\"get\"){int k;cin>>k; cout<<l.get(k)<<\"\\n\";} else{int k,v;cin>>k>>v;
l.put(k,v);} } }
```

Mock OA Checklist & Tips

1) Time allocation: 90–120 minutes for 2 coding problems; prioritize correctness then optimization.

2) Read problem carefully: note constraints, edge cases, and required output format.

3) Start with examples: run small cases on paper to confirm approach.

4) Write clean code: choose readable variable names and comment tricky parts.

5) Handle edge cases: empty inputs, single-element, duplicates, negative values.

6) Debugging: check off-by-one, index bounds, integer overflow, and null pointers.

7) Use STL effectively: vectors, unordered_map, priority_queue, deque, list.

8) Complexity check: state time & space complexity after writing solution.

9) Behavioral: prepare 4–6 STAR stories aligned with Amazon Leadership Principles.

10) After coding: run through sample tests and explain trade-offs if time permits.

Mock OA Simulation Suggestions

- Simulate 2 coding questions: one array/string medium, one tree/graph or DP medium-hard.
- 20 minutes debugging + 10 minutes for workstyle assessment.
- Use IDE/editor with fast compilation; use fast I/O in C++ (ios::sync_with_stdio(false)).
- Keep solutions concise; avoid unnecessary templating in time-limited setting.